Druid

一. Druid 是什么

Druid 是一个分布式的、支持实时多维 OLAP 分析的数据处理系统。它既支持高速的数据实时摄入处理，也支持实时且灵活的多维数据分析查询。因此 Druid 最常用的场景就是大数据背景下、灵活快速的多维 OLAP 分析。 另外，Druid 还有一个关键的特点：它支持根据时间戳对数据进行预聚合摄入和聚合分析，因此也有用户经常在有时序数据处理分析的场景中用到它。

二. 为什么用 Druid

特性

1. 亚秒响应的交互式查询，支持较高并发。
2. 支持实时导入，导入即可被查询，支持高并发导入。
3. 采用分布式 shared-nothing 的架构，可以扩展到PB级。
4. 支持聚合函数，count 和 sum，以及使用 javascript 实现自定义 UDF。
5. 支持复杂的 Aggregator，近似查询的 Aggregator 例如 HyperLoglog 以及 Yahoo 开源的 DataSketches。
6. 支持Groupby，Select，Search查询。
7. 不支持大表之间的Join，但其 lookup 功能满足和维度表的 Join。

这里最关键是前两条。

不支持

1. 不支持精确去重
2. 不支持 Join（只能进行 semi-join）
3. 不支持根据主键的单条记录更新

如果不能接受这几点，则可以考虑放弃使用 Druid 了。

三. 为什么快

数据的预聚合

Druid 可以按照给定的时间粒度和所有维度列，进行最细粒度的指标聚合运算，并加以保存为原始数据。

列式存储

对部分列进行查询时可以显著提高效率。

Bitmap 索引

利用位图对所有维度列构建索引，可以快速定位数据行。

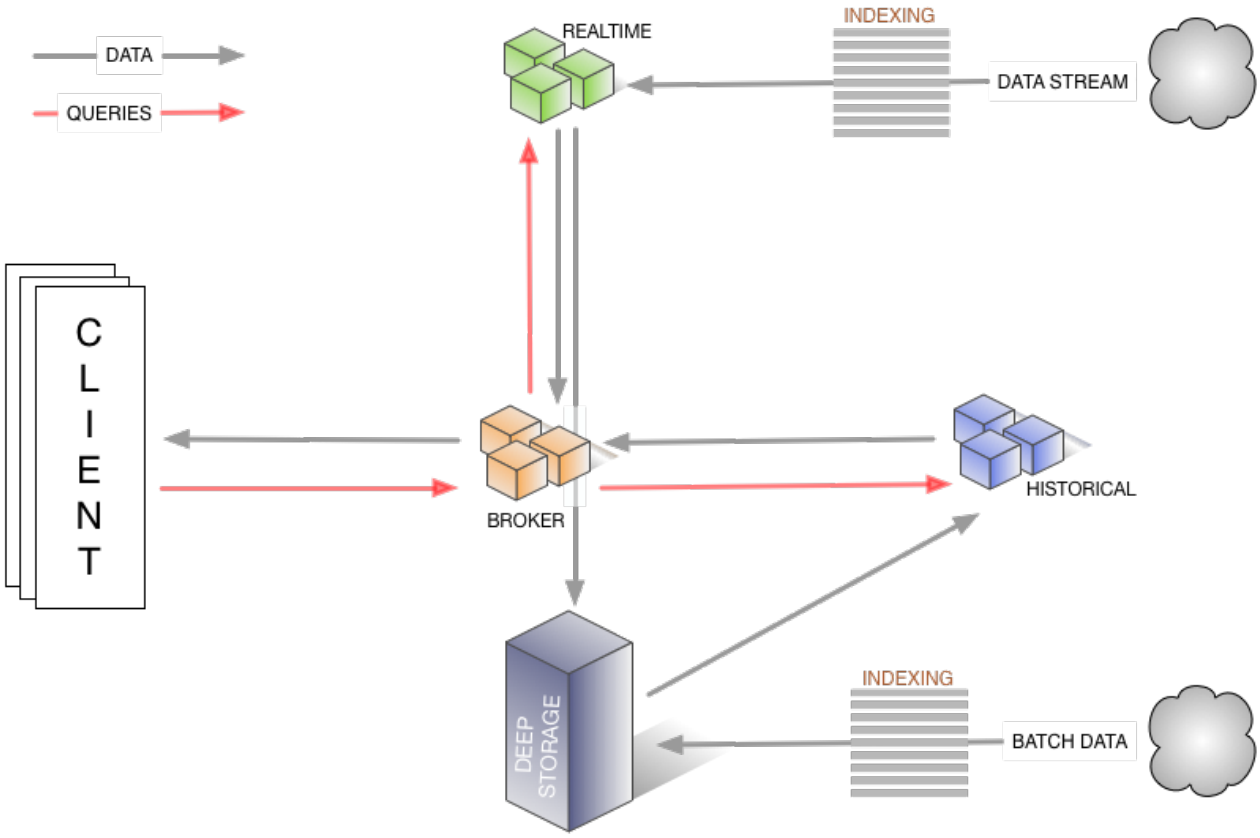
mmap

通过内存映射文件的方式加快对于 Segment 的访问。

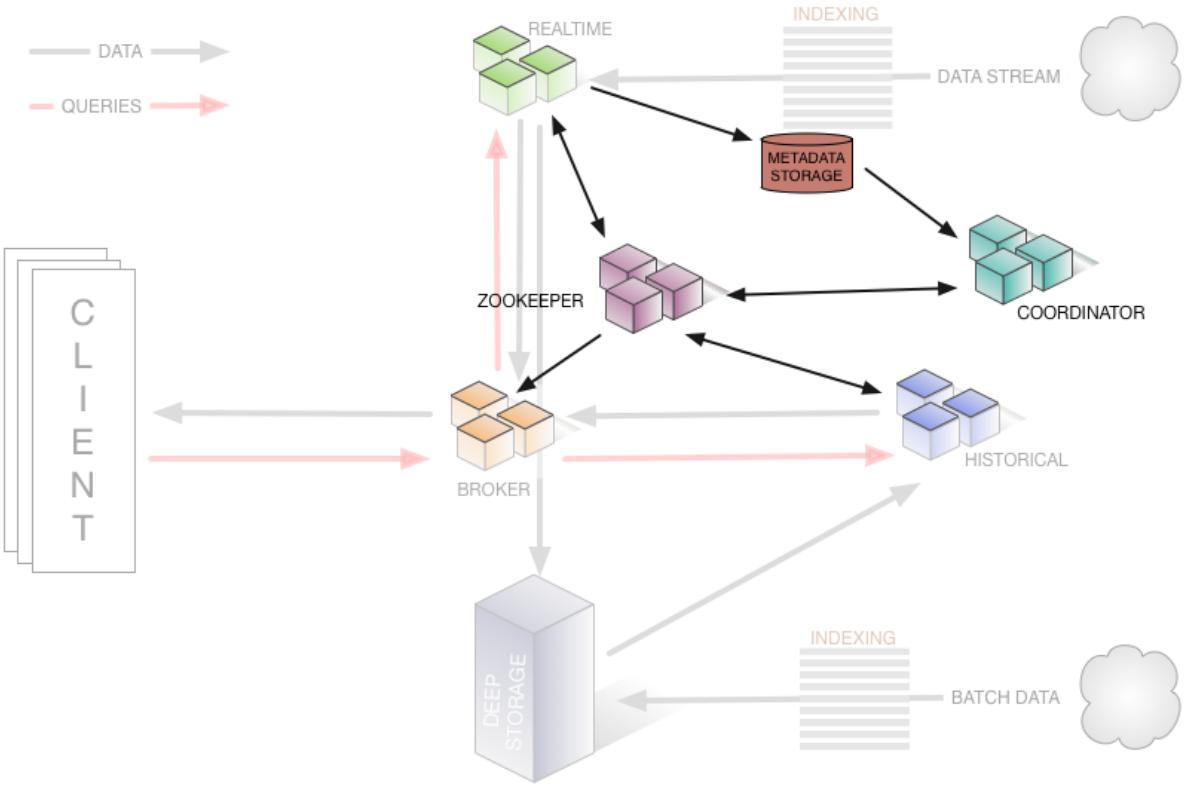
查询结果的中间缓存

支持对于查询级别和 segment 级别的缓存。

四. Druid 架构

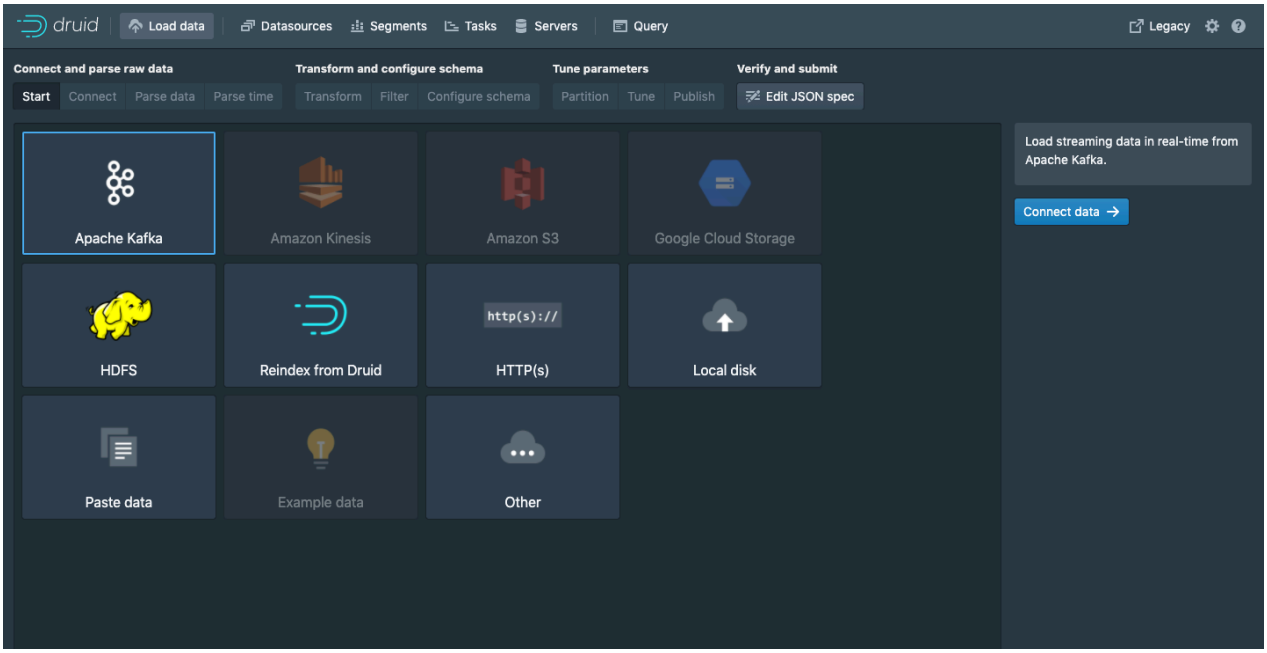


- Realtime 组件负责数据的实时摄入。
- Broker 阶段负责查询任务的分发以及查询结果的汇总，并将结果返回给用户。
- Historical 节点负责索引后的历史数据的存储，数据存储在 deep storage。Deep storage 可以是本地，也可以是HDFS 等分布式文件系统。
- Indexing service 包含两个组件（图中未画出）。
    - Overlord 组件负责索引任务的管理、分发。
    - MiddleManager 负责索引任务的具体执行。



- Zookeeper 负责存储集群的状态以及作为服务发现组件，例如集群的拓扑信息、overlord leader 的选举、indexing task 的管理等等。
- Coordinator 负责 segments 的管理，如 segments 下载、删除以及如何在 historical 之间做均衡等等。
- Metadata storage 负责存储 segments 的元信息，以及管理集群各种各样的持久化或临时性数据，例如配置信息、审计信息等等。

五. 实时数据导入



1. Start: Load data -> Apache Kafka -> Connect data

2. Connect: 配置kafka
    1. Bootstrap servers: 10.0.0.9:9092,10.0.0.8:9092
    2. Topic: ad_log_report_v3
    3. Preview
3. Parse data
4. Parse time
5. Transform
6. Filter
7. Configure schema
    1. 把account_id, unit_id, creative_id, pos_id,age 改为维度
8. Partition
9. Tune
    1. Use earliest offset: True
    2. Replicas: 3
    3. Task count: 1
    4. Task duration: PT1H
10. Publish
    1. Datasource name: ad_log_report_v3
11. Edit JSON spec
12. Run Query

```
curl -XPOST --header Content-type:application/json -d '
{"query":"SELECT * FROM (SELECT __time, count(*) FROM
\"ad_log_report_v3_test\" group by __time\n) LIMIT
100","resultFormat":"array","header":true}
' http://localhost:18082/druid/v2/sql
```

```
curl -XPOST --header Content-type:application/json -d '
{"query":"SELECT * FROM (SELECT __time, count(*) FROM
\"ad_log_report_v3_test\" group by __time\n) LIMIT
100","resultFormat":"array","header":true}
' http://druid.netlearning.tech/druid/v2/sql
```

六. 离线数据导入

```
{
    "type": "index_hadoop",
    "spec": {
        "dataSchema": {
            "dataSource": "ad_log_report_v3",
            "parser": {
                "type": "string",
                "parseSpec": {
```

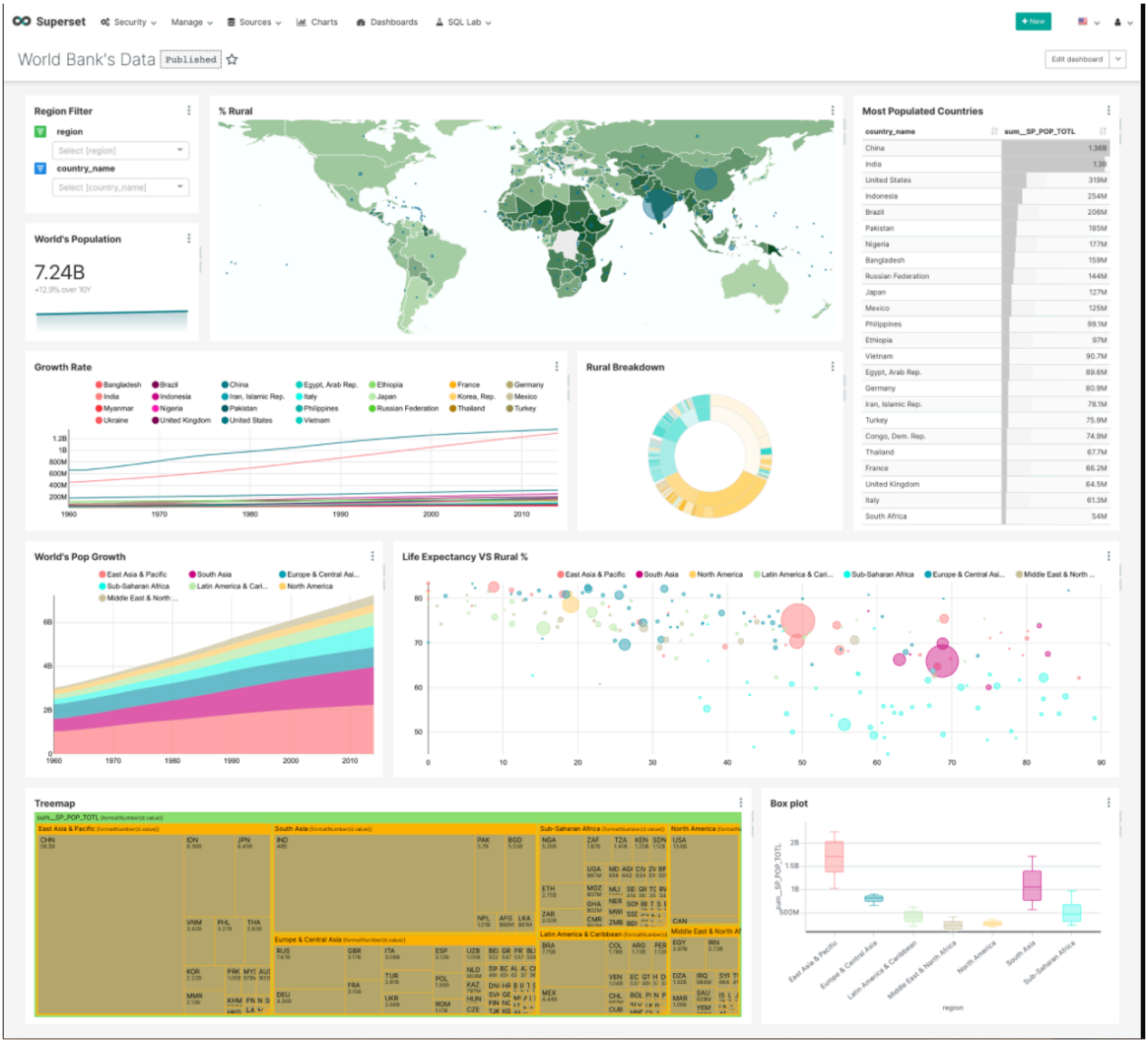
```
                "format": "json",
                "timestampSpec": {
                    "column": "timestamp",
                    "format": "millis"
                },
                "dimensionsSpec": {
                    "dimensions": [
                        "country",
                        "device_id",
                        "gender",
                        "network",
                        "os",
                        "request_id",
                        "source_type",
                        {
                            "name": "account_id",
                            "type": "string"
                        },
                        {
                            "name": "unit_id",
                            "type": "string"
                        },
                        {
                            "name": "age",
                            "type": "string"
                        },
                        {
                            "name": "creative_id",
                            "type": "string"
                        }
                    ]
                }
            }
        },
        "metricsSpec": [
            {
                "type": "count",
                "name": "count"
            },
            {
                "type": "longSum",
                "name": "sum_account_id",
                "fieldName": "account_id",
                "expression": null
            },
            {
                "type": "longSum",
                "name": "sum_age",
                "fieldName": "age",
```

```
                "expression": null
            },
            {
                "type": "longSum",
                "name": "sum_bid_price",
                "fieldName": "bid_price",
                "expression": null
            },
            {
                "type": "longSum",
                "name": "sum_click",
                "fieldName": "click",
                "expression": null
            },
            {
                "type": "longSum",
                "name": "sum_download",
                "fieldName": "download",
                "expression": null
            },
            {
                "type": "longSum",
                "name": "sum_impression",
                "fieldName": "impression",
                "expression": null
            },
            {
                "type": "longSum",
                "name": "sum_installed",
                "fieldName": "installed",
                "expression": null
            },
            {
                "type": "longSum",
                "name": "sum_pay",
                "fieldName": "pay",
                "expression": null
            },
            {
                "type": "longSum",
                "name": "sum_send",
                "fieldName": "send",
                "expression": null
            }
        ],
        "granularitySpec": {
            "type": "uniform",
            "segmentGranularity": "HOUR",
            "queryGranularity": "HOUR",
```

```
                "rollup": true,
                "intervals": [
                    "2020-10-01/2020-10-02"
                ]
            },
            "transformSpec": {
                "filter": null,
                "transforms": []
            }
        },
        "ioConfig": {
            "type": "hadoop",
            "inputSpec": {
                "type": "static",
                "paths":
"hdfs://bigdata01/user/hive/warehouse/server_log_context_hi/dt=20201001/hour=0
1"
            }
        },
        "tuningConfig": {
            "type": "hadoop"
        }
    }
}
```

Sueprset

一. Superset是什么



Superset 是一款由 Airbnb 开源的“现代化的企业级 BI（商业智能） Web 应用程序”，其通过创建和分享 dashboard，为数据分析提供了轻量级的数据查询和可视化方案。

1. 一个直观的界面来探索和可视化数据集，并创建交互式指示板。
2. 大量漂亮的可视化展示你的数据。
3. 简单、无代码、用户流可以向下钻取，并将暴露的仪表板上的数据切片和骰子。指示板和图表作为深入分析的起点。
4. 一个显示丰富的元数据浏览器的art SQL编辑器/IDE的状态，以及一个简单的工作流来创建任何结果集的可视化效果。
5. 一个可扩展的、高粒度的安全模型，允许复杂的规则对谁可以访问哪些产品特征和数据集。与主要身份验证后端集成(数据库、OpenID、LDAP、OAuth、REMOTE_USER、...)
6. 一个轻量级的语义层，允许通过定义维度和度量来控制数据源如何向用户公开。
7. 对大多数sql语言数据库的支持。
8. 与德鲁伊的深度融合允许Superset在切割和切割大型实时数据集的同时保持快速。
9. 具有可配置缓存的快速加载仪表板。

安装

安装docker

```
$ brew cask install docker

==> Creating Caskroom at /usr/local/Caskroom
==> We'll set permissions properly so we won't need sudo in the future
Password:          # 输入 macOS 密码
==> Satisfying dependencies
==> Downloading https://download.docker.com/mac/stable/21090/Docker.dmg
######################################################################## 
100.0%
==> Verifying checksum for Cask docker
==> Installing Cask docker
==> Moving App 'Docker.app' to '/Applications/Docker.app'.
&#x1f37a;  docker was successfully installed!
$ docker --version
Docker version 17.09.1-ce, build 19e2cf6
```

镜像加速

鉴于国内网络问题，后续拉取 Docker 镜像十分缓慢，我们可以需要配置加速器来解决，我使用的是网易的镜像地址：http://hub-mirror.c.163.com。

在任务栏点击 Docker for mac 应用图标 -> Perferences... -> Daemon -> Registry mirrors。在列表中填写加速器地址即可。修改完成之后，点击 Apply & Restart 按钮，Docker 就会重启并应用配置的镜像地址了。

之后我们可以通过 docker info 来查看是否配置成功。

```
$ docker info
...
Registry Mirrors:
 http://hub-mirror.c.163.com
Live Restore Enabled: false
```

Sueprset 安装

```
# clone superset的github仓库
$ git clone https://github.com/apache/incubator-superset.git
# 通过Docker compose 启动superset
$ cd incubator-superset
$ docker-compose up
# 登录 Superset
username: admin
password: admin
```

Superset 使用

1. 添加数据源：
    1. Sources -> Databases -> new
    2. Database: druid
    3. SQLAlchemy URI: druid://docker.for.mac.localhost:18082/druid/v2/sql
    4. mac下Docker容器访问宿主机端口: **docker.for.mac.localhost**
    5. Save
2. 创建数据表
3. 创建Dashboard

工具

Tmux

tmux是什么

tmux是一个 terminal multiplexer（终端复用器），它可以启动一系列终端会话。

我们使用命令行时，打开一个终端窗口，,会话开始，执行某些命令如npm run dev，关闭此终端窗口，会话结束，npm run dev服务会话随之被关闭。有时我们希望我们运行的服务如npm run dev 或者一些cd命令等，被保留，而不是关闭窗口再打开后，重新手动执行。tmux的主要用途就在于此。

它解绑了会话和终端窗口。关闭终端窗口再打开，会话并不终止，而是继续运行在执行。将会话与终端窗后彻底分离。

http://www.ruanyifeng.com/blog/2019/10/tmux.html

Chrome 插件Proxy SwitchySharp

SSH proxy

```
alias zz='ssh -p20522 root@116.213.207.93'
alias zsp="ssh -p20522 -D 7080 -f -C -q -N root@116.213.207.93"
alias z_druid='ssh -p20522 -CfNg -L 18082:10.0.0.9:18082 root@116.213.207.93'
```